Multi-Label Classification by Label Clustering based on Covariance

Reem Al-Otaibi[1,2], Meelis Kull[1], and Peter Flach[1]

[1]Intelligent System Laboratory, Computer Science, University of Bristol
Bristol, United Kingdom
[2]King Abdulaziz University, Saudi Arabia
{ra12404, meelis.kull, peter.flach}@bristol.ac.uk

Abstract. Multi-label classification is a supervised learning problem that predicts multiple labels simultaneously. One of the key challenges in such tasks is modelling the correlations between multiple labels. **LaCova** is a decision tree multi-label classifier, that interpolates between two baseline methods: Binary Relevance (BR), which assumes all labels independent; and Label Powerset (LP), which learns the joint label distribution. In this paper we introduce **LaCova-CLus** that clusters labels into several dependent subsets as an additional splitting criterion. Clusters are obtained locally by identifying the connected components in the thresholded absolute covariance matrix. The proposed algorithm is evaluated and compared to baseline and state-of-the-art approaches. Experimental results show that our method can improve the label exact-match.

Keywords: Multi-label learning; Decision trees; Covariance matrix; Splitting criteria; Clustering

1 Introduction

Classification is a learning task in which the goal is to categorise a new instance into one or more classes using the trained model. Single-label classification (binary and multiclass) is concerned with classifying a new instance into one and only one class. In binary classification, each training point can belong to one of two classes, whereas, in multiclass, the setting is more general, so that each training point can belong to one of more than two classes.

On the other hand, in multi-label classification, training points are associated with a set of classes (labels) simultaneously. There is a wide range of applications for multi-label data, such as text categorisation and image and movie tagging. For example, in a medical diagnosis, a patient may have multiple diseases at the same time.

Decision trees are one of the most popular algorithms for classification. They are built recursively by partitioning the training data. They consist of nodes: starting from the root node, the internal node represents an attribute, and the leaf node represents a class label or a probability distribution over classes. The internal node splits the training space into two or more nodes based on the chosen splitting criterion.

Among other advantages, decision trees are intuitive and comprehensible, such that the structure can be captured by inexpert users easily. Furthermore, they are selective,

which means they select the most discriminative features from the data to construct the trees, which usually number much less than the actual number of features. This has the advantage, especially in a high-dimensional feature space [11].

In fact, with regard to multi-label tasks, there are many strategies to apply the decision trees. On one hand, a single decision tree can be learnt for each label, ignoring the correlation between different labels. This approach is known as binary relevance (BR). On the other hand, a single tree can be learnt, which makes predictions for all labels together. it is known as label powerset (LP). In the work proposed in [1], authors have developed a tree based multi-label classifier using a label covariance as splitting criterion called **LaCova**. The key idea of **LaCova** is to use the label covariance matrix at every node of the tree in order to decide to treat labels independently (BR) or keep them all together (LP) and find a feature to split on.

The aim of this paper is to explore how to mediate between keeping all labels together or treating them independently, by studying ways in which the covariance matrix can suggest label clusters. Moreover, we incorporate ensemble models with **LaCova-CLus** and compare it with other ensemble algorithms.

The rest of this paper is organised as follows: Section 2 discusses the problem definition and possible approaches to learn multi-label decision trees; the **LaCova-CLus** is presented in Section 3; Section 4 gives an overview of existing approaches to introduce decision trees into multi-label classification; experimental results are presented in Section 5; and Section 6 concludes the paper.

2 Problem Definition

Before giving a formal definition, suppose the learning task is to classify a given movie into several genres so the task in this case is a multi-label classification problem. To build a decision tree for this dataset a number of methods can be used.

One of the most common approaches and perhaps the simplest way, is to build a single decision tree for each label and train it independently from the other labels' trees. So in this case, we create a tree for each movie genre. In order to do this, we need to transform the dataset into several subsets where each subset has information only about one movie category. Given a new movie, in order to predict its categories, all trees should be tested. The prediction for this movie will be the union of all binary decision trees. This approach learns number of trees equals to the number of labels, which can be hundreds or thousands in some domains.

The second approach is to learn one decision tree for all movie genres and predict all of them once. This method models the joint distributions and learns all the labels as one set. Although this approach is simple and effective, it might end up with few instances at a leaf when the number of labels is increased.

In summary, we will justify about our approach based on the advantages and disadvantages of each approach. The first method does not exploit dependencies among the labels, which is one of the main challenges in a multi-label classification [8]. Moreover, the number of learnt trees can be large, particularly in the high-dimensional space, which can be hundreds or thousands in some domains (it can be up to 174 in our experiment) . Finally, from the knowledge discovery point of view, the resulting trees of

this approach identify features relevant for one label, rather than identifying features with overall relevance [14]. In the second approach, over-fitting is an issue because it might end up with a few instances at a leaf. In addition, the exponential number of label combinations is a potential issue with this approach.

This paper proposes an algorithm that combines these two baseline approaches and produces a hybrid tree. **LaCova-CLus** uses the thresholded absolute covariance matrix in order to find the connected components among labels locally. These components are partitioned into clusters. For each cluster, we learn a single decision tree (LP).

3 The LaCova-CLus Algorithm

The area of multi-label classification has attracted many researchers. One of the main challenges is the large number of labels in real-world applications; importantly, these labels may present some correlation between them. Thus, exploiting dependencies among the labels without increasing complexity could improve the classifier performance [8,9]. Acknowledging the benefits of decision tree models, we focus our work on decision trees themselves.

In this paper, we propose **LaCova-CLus**, which is different from the previous methods as it clusters labels dynamically during the construction of the decision tree. It also tests label dependencies while growing the tree, which might change and lead to change in label clusters.

Standard decision tree algorithms use greedy searches to select splits that maximally decrease the impurity from a node to its children. There are many ways to measure the impurity of a set of instances, including entropy and Gini index. We note that the Gini index $p_j(1-p_j)$ for a binary class label j with proportion of positives p_j is the variance of a Bernoulli distribution with success probability pj. With multiple binary labels we can also consider label covariance, which for two Bernoulli variables j and k with success probabilities p_j and p_k and joint probability p_{jk} is $p_{jk} - p_j \cdot p_k$. For a set of $|L|$ labels we can form an L-by-L covariance matrix with label variances on the diagonal and pairwise covariances off the diagonal.

LaCova implemented a three-way splitting criterion, which can be summarised as follows. Firstly, if the trace of this matrix (the sum of the diagonal entries) is small, then the set of instances is nearly pure in a multi-label sense. Secondly, if the labels are actually independent, i.e., each pairwise covariance is low in magnitude, then apply BR at this point. We assessed this by calculating total absolute covariance, where the absolute value is taken to avoid positive and negative covariances cancelling out. Finally, learn all labels together and find a feature to split on. The main algorithms are given in [1].

The covariance threshold λ is required to decide whether there is a significant dependence between the labels or not. Authors in [1] derived a threshold λ that is also computed dynamically at each node of the tree. For more details on derivation λ, see [1].

The first two choices require a threshold on the sum of variances and the sum of absolute covariances, respectively. In experiments we found that, in combination with a minimum number of instances in a leaf, a variance threshold of 0 (i.e., all labels pure) works well. The covariance threshold requires a second innovation, which is presented

Algorithm 1 LaCova-CLus (D): Learn a tree-based multi-label classifier from training data.

Input: Dataset D; Labels set L, Minimum number m of instances to split
Output: Tree-based multi-label classifier
CM=Covariance Matrix
if SumOfVar$(CM) = 0$ or $|D| < m$ **then**
  Return Leaf with relative frequencies of labels
else if SumOfAbsCov$(CM) \leq \lambda$ **then**
  for each label j in D **do**
    T_j = Learn a decision tree for single label (BR) j
  end for
  Return Node with single-label decision tree T_j
else
  $clusters$=**CLUST**(L, CM)
  if $|clusters| > 1$ **then**
    for each set s in $clusters$ **do**
      T_s = Learn a decision tree for set of labels s (LP)
    end for
    Return Node with LP labels decision tree T_s
  else
    $f, \{D_i\}$ = FindBestSplit(D)
    for each child node D_i **do**
      T_i = **LaCova-CLus** (D_i)
    end for
    Return Node splitting on f with subtrees T_i
  end if
end if

in the next section. This leads to the main algorithm given in Algorithm 1, which implements the above three-way split.

In this work, we address how to mitigate between two options: learning a separate tree for each label or keep all labels together and learn one tree. The basic idea of **LaCova-CLus** is to find useful clusters using the thresholded absolute covariance matrix and decompose the set of labels into several subsets of dependent labels, build an LP classifier for each subset. It also combines BR for independent labels. In addition to the above mentioned three-way splitting criterion, **LaCova-CLus** incorporates a fourth option as follows (see Algorithm 1).

1. If the sum of variances is low, stop growing the tree.
2. Or, if the sum of absolute covariances is low, apply BR (vertical split).
3. Or, if there are label clusters, apply LP (vertical split) for each cluster (**LaCova-CLus**).
4. Or, find a good feature to split on and recurse (horizontal split).

3.1 Clustering

Algorithm 2 identifies label clusters based on the thresholded absolute covariance matrix. The first step is to assume that all labels are independent and in separate clusters. Then, it determines which pair of labels to merge in one cluster based on tunable parameter λ. The algorithm continues building the clusters by merging the clusters gradually. These clusters are generated dynamically within the tree.

Algorithm 2 CLUST(L,CM): Cluster labels based on the covariance matrix

Input: A set of labels $L = l_1, \cdots, l_{|L|}$; Covariance Matrix *CM*
Output: *n*ewClust - The final label clusters
Initialise *c*urrClust=null; *n*ewClust=null; pairList=null;
/* Build initial clusters by assuming each label is in a separate cluster. */
*c*urrClust=$\{l_1\}, \{l_2\}, \cdots, \{l_{|L|}\}$
/* Create a list of label pairs sorted in descending order of the absolute covariance value. */
pairList $\leftarrow$ sorted list
for each label pair(l_i, l_j), where $i = 0 \cdots |L|-1$ and $j = i+1 \cdots |L|$ **do**
  if all labels are in the same cluster **then**
    Stop clustering
  end if
  if AbsCov$(CM, l_i, l_j) > \lambda$ **then**
    /* Merge l_i and $l_j$to a new cluster and update the clusters. */
    *n*ewClust=*c*urrClust $\cup (l_i, l_j)$
  end if
end for
Return *n*ewClust

We use the same threshold to decide if a pair of labels are dependent or not to merge them in one cluster.

$$\lambda = \hat{\mu} + 2\hat{\sigma}$$

$$\hat{\mu} = \sqrt{\frac{2}{(n-1)\pi}} \sqrt{p_j p_k (1-p_j)(1-p_k)}$$

$$\hat{\sigma}^2 = \frac{1-\frac{2}{\pi}}{n-1} p_j p_k (1-p_j)(1-p_k)$$

where p_j and p_k are the probabilities of two labels j and k. n is the number of instances reaching a particular tree node.

4 Related Work

Many different directions have been taken in the literature to adapt decision trees for multi-label problems. We now summarise them into different approaches as follows.

A first approach transforms a multi-label problem into several binary classification tasks and builds a decision tree for each label separately which is known as BR. To classify a new instance, it outputs the union of the labels that are positively predicted by all the trees. Classifier Chains (CC) is similar to the BR concept, however, it considers labels correlation. It learns binary classifier for each label along the chain (labels ordering). Features of each classifier in the chain is incremented with the predictions of all previous classifiers along the chain [10]. Ensembles of Classifier Chains (ECC) [10] use CC as a base classifier by training a number of CC classifiers. Each classifier is trained with different order of labels in the chain and a random subset of the data.

The second method learns a single tree and predicts all the labels together. Such example is proposed by authors in [3], they adapted the C4.5 decision tree to deal with multi-label data, while the basic strategy was to define multi-label entropy over a set of multi-label examples separately. The modified entropy sums the entropies for each individual label. Although, this approach is able to identify features that are relevant to

all labels at once, splitting is not guided by label correlations. Another recent work is proposed in [6], which also builds a single tree for a multi-label dataset. They proposed a hybrid decision tree model that utilises support vector machines (SVMs) at its leaves. It is known as ML-SVMDT and it combines two models: ML-C4.5 and BR. It builds single decision trees similar to ML-C4.5, where the leaves contain BR classifiers that give multi-label predictions using SVM.

The final approach exploits the correlation between labels by applying clustering approaches. Hierarchy of multi-label classifiers (HOMER) organises all labels into a tree-shaped hierarchy with a smaller set of labels at each node [13]. In the training phase, a multi-label classifier is trained for each internal node to predict a set of labels called a meta-label. Then, it proceeds into the successor nodes of the meta-label if they belong to this set of labels. Leaf nodes construct a binary classifier to predict only a single label. Another recent work in [2] combines the LP and BR methods and is called LPBR. Its first step is to explore dependencies between labels and then to cluster these labels into several independent subsets according to the chi square χ^2 statistic. Second, a multi-label classifier is learnt: if the set contains only one label, BR is applied; otherwise, LP is used for a group of dependent labels.

5 Experimental Evaluation

LaCova, **LaCova-CLus** and ML-C4.5 have been implemented in Java using Meka[1]. Meka was used for BR, LP, and CC, whereas Mulan[2] was used for the LPBR algorithm. In all these algorithms, the trees are produced by J48 algorithm.

Initially, we performed experiments and compared **LaCova-CLus** to other baseline approaches: BR and LP. Then, we evaluated and compared the proposed model and its ensemble version to other state-of-the-art multi-label algorithms.

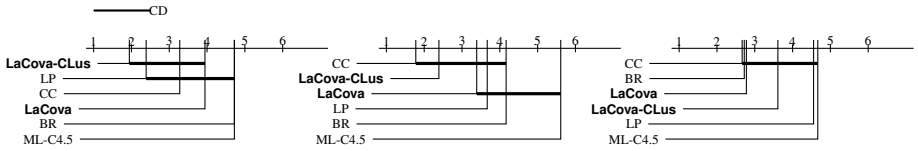
LPBR needs parameters configuration such as non-improving counter to stop clustering. The default parameters setting in Mulan were 10 for non-improving counter and 10-fold cross validation for testing the clustering performance. For large datasets it takes days to run the experiments and then cause out of memory error. Therefore, these parameters were set to 5 for both non-improving counter and 5-fold cross validation for clusters evaluation.

Nevertheless, the largest three datasets in terms of the number of labels: CAL500, Language log and Enron, LPBR takes hours to execute and then reports out of memory problem (as shown in Table 1). We report the results on others at least to see how it performs but we did not include its results in the significant test.

Considering ensemble models, all methods involve bootstrap sampling the training set. Ensemble iterations are set to 10. The predictions are averaged per label to produce the classifications confidence values.

We evaluate the algorithms on 9 commonly used benchmarks from the Meka and Mulan repositories. We used the most common multi-label metrics, namely multi-label

[1] http://meka.sourceforge.net/
[2] http://mulan.sourceforge.net/



(a) exact-match (b) hamming loss (c) f-measure

Fig. 1: Critical Difference diagrams using pairwise comparisons for those experiments where the Friedman test gives significance at 0.05. It considers only single classifier algorithms.

accuracy (Jaccard index), exact-match (subset accuracy), hamming loss and micro averaged f-measure [5,7,12]. Tables 1 and 2 show the average 10-fold cross validation of the single classifier and ensembles, respectively.

Table 1 also shows the average rank of each approach. Both **LaCova** and **LaCova-CLus** have the best average rank in terms of the multi-label accuracy. **LaCova** wins in five datasets out of nine, whereas **LaCova-CLus** wins in two datasets out of nine. In relation to exact-match, **LaCova-CLus** has the best average rank followed by LP, which suggests that clustering labels into dependent subsets may improve the exact-match. CC has the best average rank for both hamming loss and f-measure. **LaCova-CLus** and **LaCova** have the second best average rank for hamming loss and f-measure, respectively.

We conducted the Friedman test based on the average ranks for all datasets in order to verify whether the differences between algorithms are statistically significant [4]. For exact-match, hamming loss, f-measure the Friedman test gave a significant difference at 5% confidence so we proceed to a post-hoc analysis based on Nemenyi statistics as shown in Figure 1. Regarding the ensemble models, there is no significant difference between the algorithms.

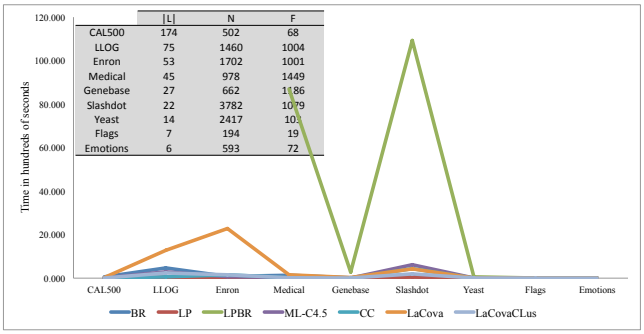


Fig. 2: Comparison of build time in hundreds of seconds. The table shows statistics of the datasets used in the experiments. $|L|$ is the number of labels, F is the number of features and N is the number of examples.

Table 1: Average 10-fold cross validation on 9 datasets comparing single classifier algorithms. The lower value of hamming loss is the better, and for the other metrics the higher value is the better.

	BR	LP	LPBR	ML-C4.5	CC	**LaCova**	**LaCova-CLus**
Multi-label Accuracy							
CAL500	0.201	0.204	-	0.225	0.208	**0.226**	0.206
LLOG	0.245	0.234	-	0.208	**0.247**	0.100	0.243
Enron	**0.296**	0.277	-	0.344	**0.315**	**0.315**	0.294
Medical	0.729	0.724	0.725	0.541	0.728	0.548	**0.736**
Genebase	0.901	0.980	0.955	0.922	**0.986**	0.918	0.982
Slashdot	0.425	0.420	0.376	0.312	0.429	0.444	**0.465**
Yeast	0.391	0.396	0.411	0.414	0.218	**0.431**	0.397
Flags	0.536	0.550	0.541	0.527	0.531	**0.574**	0.565
Emotions	0.402	0.416	0.417	0.411	0.388	**0.425**	0.405
average rank	4.3	4.1	-	4.1	3	**2.6**	2.8
exact-match							
CAL500	0	0	-	0	0	0	0
LLOG	0.184	0.205	-	0.172	0.203	0.063	**0.211**
Enron	0.022	**0.079**	-	0.064	0.076	0.058	0.077
Medical	0.607	0.642	0.655	0.389	0.641	0.393	**0.661**
Genebase	0.809	0.962	0.937	0.882	**0.971**	0.844	0.968
Slashdot	**0.425**	0.368	0.306	0.064	0.356	0.351	0.411
Yeast	0.035	**0.134**	0.122	0.096	0.118	0.120	0.122
Flags	0.098	0.139	0.145	0.108	0.150	0.160	**0.187**
Emotions	0.125	0.202	0.165	0.165	0.157	**0.219**	0.187
average rank	4.7	2.3	-	4.7	3.2	3.9	**1.9**
hamming loss							
CAL500	0.223	0.201	-	0.213	**0.189**	0.214	0.201
LLOG	0.031	0.026	-	0.033	**0.023**	0.035	0.025
Enron	0.082	0.078	-	0.084	**0.069**	0.075	0.076
Medical	0.013	0.014	0.017	0.025	**0.011**	0.025	0.013
Genebase	0.008	0.002	0.019	0.014	**0.001**	0.011	0.002
Slashdot	0.055	0.058	0.086	0.084	**0.045**	0.057	0.054
Yeast	0.296	0.288	0.363	0.297	0.307	**0.272**	0.285
Flags	0.317	0.297	0.309	0.334	0.295	**0.281**	0.285
Emotions	0.296	0.304	0.309	0.323	0.305	**0.287**	0.295
average rank	4.1	3.6	-	5.6	**1.7**	3.3	2.3
f-measure							
CAL500	0.334	0.332	-	0.363	0.338	**0.364**	0.334
LLOG	0.175	0.122	-	0.103	**0.177**	0.121	0.136
Enron	**0.428**	0.364	-	0.426	0.014	0.421	0.391
Medical	0.778	0.744	0.773	0.587	**0.786**	0.618	0.758
Genebase	0.921	0.982	0.853	0.870	**0.988**	0.896	0.981
Slashdot	0.506	0.429	0.381	0.416	**0.512**	0.476	0.473
Yeast	0.541	0.522	0.531	0.546	0.528	**0.560**	0.526
Flags	0.686	0.689	0.686	0.674	0.677	**0.711**	0.707
Emotions	0.539	0.521	0.540	0.523	0.499	**0.544**	0.527
average rank	2.7	4.5	-	4.6	**2.6**	2.7	3.6

5.1 Summary

The general conclusion of these experiments, which compare **LaCova** and **LaCova-CLus** with other different algorithms, are summarised in the following points:

- There is no algorithm that performs well in all evaluation measures. Multi-label classifiers can be selected depending on the dataset and the desired evaluation metrics.
- Classifiers that transform multi-label problems into several binary problems are good for both hamming loss and f-measure. A good example for this is binary relevance approach.

Table 2: Average 10-fold cross validation on 9 datasets comparing ensemble algorithms (10 iterations).

	Multi-label Accuracy		
	ECC	**LaCova**	**LaCova-CLus**
CAL500	0.282	**0.288**	0.249
LLOG	**00.281**	0.084	0.084
Enron	**0.388**	0.384	0.372
Medical	**0.773**	0.439	0.744
Genebase	**0.974**	0.919	0.972
Slashdot	0.466	0.446	**0.484**
Yeast	0.505	**0.519**	0.496
Flags	0.565	0.579	**0.587**
Emotions	0.493	**0.500**	0.466
average rank	**01.66**	2.05	2.27

	Exact-match		
	ECC	**LaCova**	**LaCova-CLus**
CAL500	0	0	0
LLOG	**0.199**	0.003	0.003
Enron	0.047	0.029	**0.055**
Medical	**0.671**	0.269	0.635
Genebase	**0.953**	0.844	0.949
Slashdot	0.320	0.341	**0.359**
Yeast	0.124	**0.159**	0.120
Flags	0.135	0.129	**0.187**
Emotions	**0.226**	0.261	**0.226**
average rank	**1.83**	2.27	1.88

	hamming loss		
	ECC	**LaCova**	**LaCova-CLus**
CAL500	0.209	**0.167**	0.193
LLOG	**0.033**	0.037	0.037
Enron	0.068	**0.060**	0.071
Medical	**0.011**	0.025	0.013
Genebase	**0.002**	0.014	0.003
Slashdot	**0.051**	0.061	0.056
Yeast	0.239	**0.213**	0.245
Flags	0.294	0.271	**0.263**
Emotions	0.258	**0.234**	0.254
average rank	**01.88**	1.94	2.16

	f-measure		
	ECC	**LaCova**	**LaCova-CLus**
CAL500	0.435	**0.442**	0.394
LLOG	**0.219**	0.126	0.126
Enron	0.519	**0.529**	0.502
Medical	**0.813**	0.570	0.772
Genebase	**0.980**	0.873	0.968
Slashdot	**0.548**	0.523	0.512
Yeast	0.634	**0.648**	0.627
Flags	0.707	0.719	**0.726**
Emotions	0.610	**0.622**	0.590
average rank	**1.66**	1.83	2.5

- There are some proposed solutions which combines binary relevance because of its effectiveness for the above mentioned metrics. To name a few, classifier chain and **LaCova**. We can see from the experiments that these methods have good results for hamming loss and f-measure.
- Exact-match is a strict measure. Considering correlation between labels can get higher exact-match. **LaCova-CLus** and LP achieve better exact-match. CC also considers label correlation, however, it depends on the labels order. For that reason, they propose ensemble of classifier chain thats tries different label orders.
- **LaCova** and **LaCova-CLus** have better multi-label accuracy among others.
- In case of high dimensional space in terms of number of labels, features, examples, **LaCova-CLus** is much faster than **LaCova**. Figure 2 shows the build time for all algorithms across datasets.

6 Conclusion

In this paper we have presented a novel algorithm for multi-label classification called **LaCova-CLus**. The key idea of this algorithm is to compute label covariance matrix at each node of the tree in order to measure the correlation and cluster labels dynamically. It interpolates between two well-known multi-label classifiers: LP and BR by introducing four splitting ways that enables local decisions.

To evaluate **LaCova-CLus** we first compared it to baseline and other state-of-the-art approaches. Then, we evaluated its ensemble to other ensemble methods. We used four common evaluation metrics and nine datasets. Experimental results indicate that it outperforms these methods for exact-match on the average rank.

Acknowledgment

Reem is a PhD student who is sponsored by King Abdulaziz University, Saudi Arabia. This work was partly supported by the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences and Technologies ERA-Net (CHISTERA), and funded by the Engineering and Physical Sciences Research Council in the UK under grant EP/K018728/1.

References

1. Al-Otaibi, R., Kull, M., Flach, P.: Lacova: A tree-based multi-label classifier using label covariance as splitting criterion. In: Proceedings of the IEEE 13th International Conference on Machine Learning and Application (ICMLA-2014). pp. 74–79. IEEE, Detroit, USA (December 2014)
2. Chekina, L., Gutfreund, D., Kontorovich, A., Rokach, L., Shapira, B.: Exploiting label dependencies for improved sample complexity. Machine Learning 91(1), 1–42 (Apr 2013)
3. Clare, A., Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: In: Lecture Notes in Computer Science. pp. 42–53. Springer (2001)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (Dec 2006)
5. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: Proceedings of the 14th ACM international conference on Information and knowledge management. pp. 195–200. CIKM '05, ACM, New York, NY, USA (2005)
6. Gjorgjevikj, D., Madjarov, G., Dzeroski, S.: Hybrid decision tree architecture utilizing local svms for efficient multi-label learning. IJPRAI 27(7) (2013)
7. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 22–30. Springer (2004)
8. Luaces, O., Díez, J., Barranquero, J., del Coz, J.J., Bahamonde, A.: Binary relevance efficacy for multilabel classification. Progress in AI 1(4), 303–313 (2012)
9. Read, J., Martino, L., Luengo, D.: Efficient monte carlo optimization for multi-label classifier chains. In: Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (2013)
10. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II. pp. 254–269. ECML PKDD '09, Springer-Verlag, Berlin, Heidelberg (2009)
11. Rokach, L., Maimon, O.: Data Mining with Decision Trees: Theory and Applications. World Scientific Publishing Co., Inc., River Edge, NJ, USA (2008)
12. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. In: Machine Learning. pp. 297–336 (1999)
13. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In: ECML/PKDD 2008 Workshop on Mining Multidimensional Data (2008)
14. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Mach. Learn. 73(2), 185–214 (Nov 2008)